

Appium

Nombre: Fabián Fernández

Introducción

Appium es una herramienta de código abierto para automatizar pruebas en aplicaciones nativas, web móviles e híbridas en plataformas móviles iOS, móviles Android y Windows. Las aplicaciones nativas son aquellas escritas con los SDK de iOS, Android o Windows. Las aplicaciones web móviles son aplicaciones web a las que se accede mediante un navegador móvil. Las aplicaciones híbridas tienen un envoltorio alrededor de una "vista web", un control nativo que permite la interacción con el contenido web. Proyectos como Apache Córdova facilitan la creación de aplicaciones utilizando tecnologías web que luego se empaquetan en un contenedor nativo, creando una aplicación híbrida.

Appium es "multiplataforma": le permite escribir pruebas en múltiples plataformas (iOS, Android, Windows), utilizando la misma API. Esto permite la reutilización de código entre los tests de pruebas de iOS, Android y Windows.

Idea General

Appium es un servidor que expone una API REST, el cual se ejecuta y se mantiene escuchando hasta que un cliente genere una conexión. Este cliente a través de ciertos parámetros, le indica que tipo de sesión de automatización iniciar y que comportamientos de automatización implementar, una vez iniciada una sesión.

Una vez el cliente este conectado al servidor Appium, este ejecutará comandos los cuales serán enviados a el servidor Appium y este ejecutará los comandos en un dispositivo móvil ya sea virtual o real para luego responder al cliente con una http response que representa la ejecución del comando.

La automatización siempre se realiza en el contexto de una sesión. Los clientes inician una sesión con un servidor de formas específica para cada biblioteca, pero todos terminan enviando una solicitud POST al servidor, con un objeto JSON llamado objeto de 'Capabilities'. Este objeto es el cual le indica al servidor que tipo de sesión iniciar.

Instalación

Para instalar Appium en Windows 10 es necesario lo siguiente:

Ingresa a la página de <https://nodejs.org/en/download/> y descarga el instalador de 64 bits para instalar Node.js y el gestor de paquetes npm.

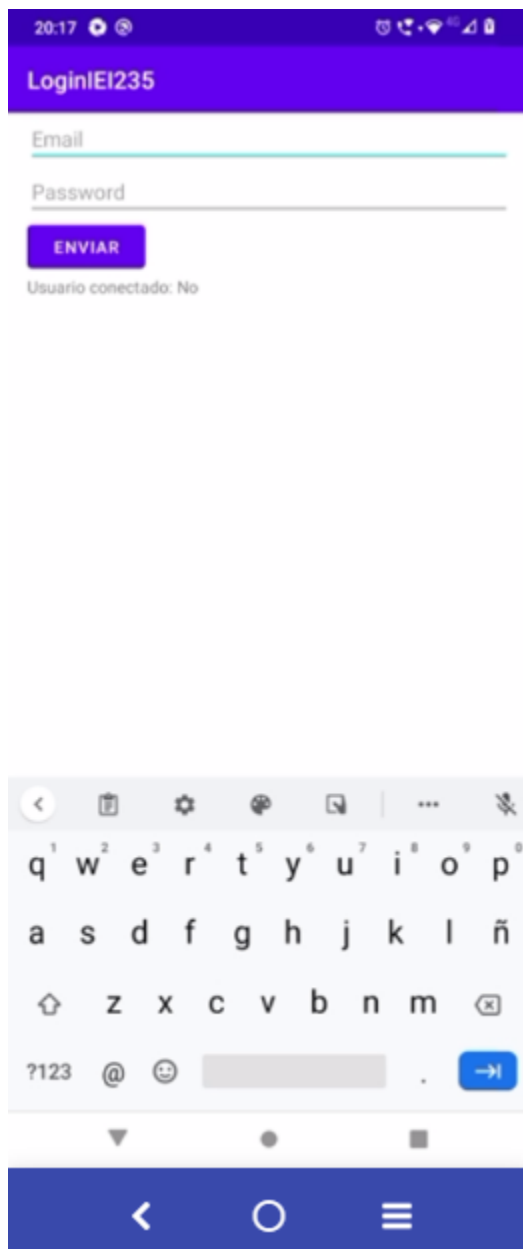
Utiliza npm para instalar Appium utilizando el comando:

- `npm install -g appium`

Configura el controlador específico para la aplicación a automatizar. Para el caso de este documento se automatizará la prueba de casos en una aplicación Android nativa por lo que es necesario utilizar UiAutomator2 Driver. Para más información de como configurar este controlador ingresa a <http://appium.io/docs/en/drivers/android-uiautomator2/index.html>

Aplicación de ejemplo

Como se indico anteriormente se creó una aplicación nativa de Android para probar una serie de casos de prueba utilizando la herramienta de Appium. Esta aplicación consiste en un sistema de login compuesto por el email identificador del usuario, su contraseña y un botón para enviar la información antes mencionada.



The screenshot shows a mobile application interface for login. At the top, a status bar displays the time 20:17 and various system icons. Below this is a purple header bar with the text "LoginIEI235". The main content area is white and contains two input fields: "Email" and "Password", each with a light blue underline. Below the password field is a purple button labeled "ENVIAR". Underneath the button, the text "Usuario conectado: No" is displayed. At the bottom of the screen, a virtual keyboard is visible, showing letters, numbers, and symbols. The bottom of the image shows a blue navigation bar with three icons: a back arrow, a home circle, and a menu hamburger icon.

Foto de vista "Login" aplicación

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_weight="1"
        android:paddingLeft="16dp"
        android:paddingRight="16dp"
        android:orientation="vertical">
        <EditText
            android:id="@+id/Email"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:inputType="textEmailAddress"
            android:hint="Email"/>
        <EditText
            android:id="@+id/Password"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:inputType="textPassword"
            android:hint="Password"/>
        <Button
            android:id="@+id/botonEnviar"
            android:layout_width="100dp"
            android:layout_height="wrap_content"
            android:layout_gravity="left"
            android:onClick="EnviarData"
            android:text="Enviar" />
        <TextView
            android:id="@+id/estadoUsuarioConectado"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="Usuario conectado: No"/>
    </LinearLayout>
</androidx.constraintlayout.widget.ConstraintLayout>
```

Foto del código del layout utilizado en la vista de “Login”

Dentro de la lógica de la aplicación esta comprueba si los datos de email y contraseña ingresados están dentro de la memoria al hacer click en el botón.

Dentro de la memoria solo se tiene el usuario con email:

[“fabian.fernandez@sansano.usm.cl”](mailto:fabian.fernandez@sansano.usm.cl) y con contraseña “test”.

Creación de cliente Appium y casos de prueba:

Para este documento se escogió crear un cliente en JavaScript. Para esto se debe crear un directorio y dentro de este se debe ejecutar el comando:

- npm init -y

Una vez inicializado es necesario instalar la librería con la cual se conectará el cliente con el servidor Appium ejecutando el siguiente comando dentro del mismo directorio:

- npm install webdriverio

Ahora es necesario crear un archivo que será llamado index.js con la siguiente información:

```
1  const wdio = require("webdriverio");
2  const assert = require("assert");
3
4  const opts = {
5    path: '/wd/hub',
6    port: 4723,
7    capabilities: {
8      platformName: "Android",
9      platformVersion: "10",
10     deviceName: "Android Device",
11     app: "C:/Users/MisterX/Desktop/Nueva carpeta/LoginIEI235-debug.apk",
12     appPackage: "com.example.loginiei235",
13     appActivity: ".MainActivity",
14     automationName: "UiAutomator2"
15   }
16 };
```

- La variable "wdio" sirve para importar la librería "webdriverio"
- La variable "opts" corresponden a los argumentos que serán enviados al servidor de Appium para configurar la sesión.
- En el JSON capabilities se señalan los parámetros necesarios para ejecutar la aplicación correctamente.
 - El conjunto mínimo de capacidades requeridas para cualquier controlador de Appium debe incluir:
 - platformName: el nombre de la plataforma para automatizar
 - platformVersion: la versión de la plataforma para automatizar
 - deviceName: el tipo de dispositivo a automatizar
 - app: la ruta a la aplicación que desea automatizar
 - AutomationName: el nombre del controlador que desea utilizar.

- Para más información consulte: <http://appium.io/docs/en/about-appium/getting-started/?lang=es>

```
18 async function main () {
19   const client = await wdio.remote(opts);
20   let Elementos = await client.$$('android.widget.EditText');
21   let boton = await client.$('android.widget.Button');
22   let textos = await client.$$('android.widget.TextView');
23   let estadoUsuario = textos[1];
24   let estado = await estadoUsuario.getText();
25   let email = Elementos[0];
26   let password = Elementos[1];
27   //prueba1
28   await email.setValue("fabian.fernandez@sansano.usm.cl");
29   await password.setValue("test");
30   await boton.click();
31   estado = await estadoUsuario.getText();
32   assert.strictEqual(estado,"Usuario conectado: Si")
33   //prueba2
34   await email.setValue("test@test.cl");
35   await password.setValue("test");
36   await boton.click();
37   estado = await estadoUsuario.getText();
38   assert.strictEqual(estado,"Usuario conectado: No")
39   //prueba3
40   await email.setValue("");
41   await password.setValue("");
42   await boton.click();
43   estado = await estadoUsuario.getText();
44   assert.strictEqual(estado,"Usuario conectado: No")
45
46   await client.deleteSession();
47 }
48
49 main();
```

- La línea 19 sirve para conectar el cliente con el servidor de appium usando la variable "opts"
- Desde la línea 20 a 22 sirve para identificar elementos dentro de la vista del login
- Desde la línea 28 hasta 32 se realiza una prueba que consiste en asignarles un valor a los parámetros email y password para luego ejecutar un click del botón enviar de la aplicación. Finalmente se ejecuta una asertion en la cual se comprueba si el usuario realmente logró conectar al sistema.

```

1  const wdio = require("webdriverio");
2  const assert = require("assert");
3
4  const opts = {
5    path: '/wd/hub',
6    port: 4723,
7    capabilities: {
8      platformName: "Android",
9      platformVersion: "10",
10     deviceName: "Android Device",
11     app: "C:/Users/MisterX/Desktop/Nueva carpeta/LoginIEI235-debug.apk",
12     appPackage: "com.example.loginiei235",
13     appActivity: ".MainActivity",
14     automationName: "UiAutomator2"
15   }
16 };
17
18 async function main () {
19   const client = await wdio.remote(opts);
20   let Elementos = await client.$$('android.widget.EditText');
21   let boton = await client.$('android.widget.Button');
22   let textos = await client.$$('android.widget.TextView');
23   let estadoUsuario = textos[1];
24   let estado = await estadoUsuario.getText();
25   let email = Elementos[0];
26   let password = Elementos[1];
27   //prueba1
28   await email.setValue("fabian.fernandez@sansano.usm.cl");
29   await password.setValue("test");
30   await boton.click();
31   estado = await estadoUsuario.getText();
32   assert.strictEqual(estado, "Usuario conectado: Si")
33   //prueba2
34   await email.setValue("test@test.cl");
35   await password.setValue("test");
36   await boton.click();
37   estado = await estadoUsuario.getText();
38   assert.strictEqual(estado, "Usuario conectado: No")
39   //prueba3
40   await email.setValue("");
41   await password.setValue("");
42   await boton.click();
43   estado = await estadoUsuario.getText();
44   assert.strictEqual(estado, "Usuario conectado: No")
45
46   await client.deleteSession();
47 }
48
49 main();
50

```

Foto de archivo index.js

Los casos utilizados en la aplicación fueron los siguientes:

Descripción	Email	Password	Resultado esperado
-------------	-------	----------	--------------------

En caso de que un usuario existente en el sistema intente ingresar	fabian.fernandez@sansano.usm.cl	test	Usuario conectado: Si
En caso de que un usuario que no pertenece al sistema intente ingresar	test@test.cl	test	Usuario conectado: No
En caso de que se ingrese algún elemento vacío			Usuario conectado: No

Como ejecutar Appium:

Para ejecutar appium por consola se debe ejecutar el comando:

- appium

Luego se debe acceder al directorio donde se dejó el archivo index.js y se debe ejecutar el comando:

- node index.js

Si el dispositivo móvil está conectado por usb y todo está configurado correctamente el cliente generara la conexión con el servidor appium y ambos empezará a mostrar logs, tanto el cliente como el servidor. Finalmente, en el servidor se indicará si el test fallo o si se ejecutó con éxito.

Ventajas:

Una de las ventajas que presenta Appium es el hecho de que consiste un servidor API REST. Esto implica que puede ser utilizado desde cualquier lenguaje que utilice un http client.

Appium es soportado por librería de clientes en los siguientes lenguajes: Java, Ruby, Python, PHP, JavaScript, and C#

Es compatible con cualquier framework de pruebas y casos de pruebas usados desde alguno de los lenguajes disponibles para appium cliente.

Permite automatizar pruebas para aplicaciones móviles nativas de Android e IOS, webs móviles, híbridas y aplicaciones de escritorio de Windows.

Desventajas:

La documentación que posee es insuficiente con ejemplos poco explícitos.

Como la herramienta muestra tantos logs en la consola identificar los realmente importantes se dificulta.

Debido a que consiste en un corredor de pruebas, en caso de intentar realizar testing unitario por ejemplo, se necesita de algún framework externo que le permita generar esa lógica en el cliente appium.

Conclusión:

Appium es una herramienta que sirve para automatizar pruebas en distintos tipos de aplicaciones móviles ya sea nativa, híbridas y web. Para esto utiliza una arquitectura de servidor-cliente. Esta arquitectura le permite poder generar pruebas en distintos tipos de lenguajes a través de librerías que le

permiten conectar con el servidor a través de una API REST. Finalmente concluye que es una herramienta bastante útil para el testing en aplicaciones móviles.